###### МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

###### ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

###### НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

###### Факультет информационных технологий

**Кафедра параллельных вычислений**

ОТЧЕТ

О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

«ВЫСОКОУРОВНЕВАЯ РАБОТА С ПЕРИФЕРИЙНЫМИ УСТРОЙСТВАМИ»

студента 2 курса, 22202 группы

**Бальчинова А.С.**

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

В.А. Перепёлкин

Новосибирск 2023

**СОДЕРЖАНИЕ**

[ЦЕЛЬ 3](#_Toc18443921)

[ЗАДАНИЕ 3](#_Toc18443922)

[ОПИСАНИЕ РАБОТЫ](#_Toc18443923) 3-6

[ЗАКЛЮЧЕНИЕ 6](#_Toc18443924)

# ЦЕЛИ

1. Ознакомиться с программированием периферийных устройств на примере ввода данных с Web-камеры с использованием библиотеки OpenCV.

# ЗАДАНИЕ

1. Реализовать программу с использованием OpenCV, которая получает поток видеоданных с камеры и выводит его на экран.

2. Выполнить произвольное преобразование изображения.

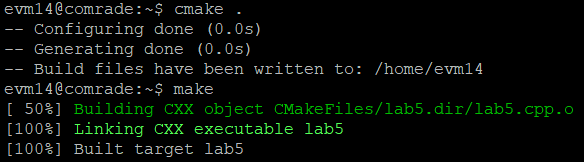
3. Измерить количество кадров, обрабатываемое программой в секунду. Оценить долю времени, затрачиваемого процессором на обработку (ввод, преобразование, показ) видеоданных, получаемых с камеры.

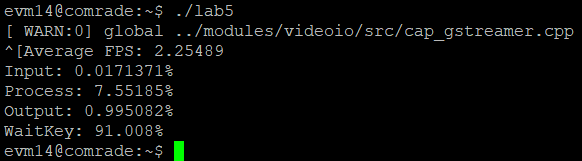
# ОПИСАНИЕ РАБОТЫ

1. Реализация программы на языке программирования C++.

#include <iostream>  
#include <chrono>  
#include <opencv2/core.hpp>  
#include <opencv2/highgui.hpp>  
#include <opencv2/videoio.hpp>  
using namespace std;  
using namespace cv;  
  
//Инверсия кадра.  
void invert(Mat& img) {  
 for (int i = 0; i < img.rows; i++) {  
 for (int j = 0; j < img.cols; j++) {  
 cv::Vec3b& pixel = img.at<cv::Vec3b>(i, j);  
 pixel[0] = 255 - pixel[0];  
 pixel[1] = 255 - pixel[1];  
 pixel[2] = 255 - pixel[2];  
 }  
 }  
}  
  
int main() {  
 int frames = 1000; //Количество кадров, обрабатываемых программой  
 int input = 0;  
 int process = 0;  
 int output = 0;  
 int time = 0;  
 int wait = 0;  
 //Создаётся поток ввода видеоданных с первой камеры (нумерация начинается с нуля).  
 //Результат операции – дескриптор обрабатываемого потока видеоданных.  
 VideoCapture capture(0);  
 if (!capture.isOpened()) {  
 return 0;  
 }  
 //Создаётся объект матрицы, в которую будем складывать кадр в цикле.  
 Mat frame;  
 int framesLeft = frames;  
 while (0 < framesLeft--) {  
 //Засекаем начало итерации.  
 auto iterStart = chrono::system\_clock::now();  
 //На каждой итерации извлекаем кадр из открытого потока ввода видеоданных.  
 //В image кладём дескриптор матрицы растрового изображения,  
 //которое будет содержать один кадр из потока видеоданных.  
 capture.read(frame);  
 // Засекаем конец загрузки кадра.  
 auto iterInput = chrono::system\_clock::now();  
 //Преобразование кадра.  
 invert(frame);  
 //Засекаем конец обработки кадра.  
 auto iterProcess = chrono::system\_clock::now();  
 //Текущий кадр выводим в окно с именем Image  
 imshow("Image", frame);  
 //Засекаем конец вывода кадра.  
 auto iterOutput = chrono::system\_clock::now();  
 //Ожидаем нажатие Escape от пользователя.  
 char c = waitKey(30);  
 if (c == 27) {  
 break;  
 }  
 //Засекаем конец итерации.  
 auto iterEnd = chrono::system\_clock::now();  
 //Вычисляем время, затраченное на каждый из процессов.  
 input += chrono::duration\_cast<std::chrono::milliseconds>(iterInput - iterStart).count();  
 process += chrono::duration\_cast<std::chrono::milliseconds>(iterProcess - iterInput).count();  
 output += chrono::duration\_cast<std::chrono::milliseconds>(iterOutput - iterProcess).count();  
 time += chrono::duration\_cast<std::chrono::milliseconds>(iterEnd - iterStart).count();  
 wait += chrono::duration\_cast<std::chrono::milliseconds>(iterEnd - iterOutput).count();  
 }  
 capture.release();  
 destroyWindow("Image");  
 //Выводим полученные данные.  
 double inp = (double)input \* 100 / time;  
 double prc = (double)process \* 100 / time;  
 double out = (double)output \* 100 / time;  
 double frm = (double)(frames – framesLeft) \* 1000 / time;  
 double wtk = (double)wait \* 100 / time;  
 cout << "Average FPS: " << frm << endl;  
 cout << "Input: " << inp << "%" << endl;  
 cout << "Process: " << prc << "%" << endl;  
 cout << "Output: " << out << "%" << endl;  
 cout << "WaitKey: " << wtk << "%" << endl;  
 return 0;  
}

1. Запуск программы и результат.

****

****

Меньше всего времени тратится на ввод данных, вывод также почти не занимает времени, относительно обработки изображения.

Большая часть времени тратится на waitKey (приблизительно 91%).

**Изображение до обработки**

****

**Изображение после обработки**

****

# ЗАКЛЮЧЕНИЕ

Ознакомились с программированием периферийных устройств на примере ввода данных с Web-камеры с использованием библиотеки OpenCV.